ARMY RESEARCH LABORATORY

Microelectromechanical System (MEMS) Switch Test

by Stanley Karter and Tony Ivanov

ARL-TR-5439 **January 2011**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Adelphi, MD 20783-1197

ARL-TR-5439 **January 2011**

Microelectromechanical System (MEMS) Switch Test

Stanley Karter and Tony Ivanov
Sensors and Electron Devices Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
January 2011	Final	July 2010

4. TITLE AND SUBTITLE
Microelectromechanical System (MEMS) Switch Test

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Stanley Karter and Tony Ivanov

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-SER-E
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-5439

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Analyzing microelectromechanical system (MEMS) switch behavior is a new field of study, which guides the production of MEMS switches. I analyzed switch behavior using a digital circuit synthesized on a field-programmable gate array (FPGA) chip. The chip had two outputs and one input connected to a MEMS switch and a radio frequency (RF) generator. It opened and closed the switch repeatedly and measured the resistance across that switch. The circuit consisted of the switch, a resistor, and the voltage out signal. I calculated the switch resistance from the voltage output applied on one end of the switch and the voltage input that lies between the switch and the resistor. The second output was the bias voltage needed to close the switch. A DA/AD converter translated the digital signal from the chip to the analog signal interacting with the switch. Parameters, such as the bias voltage, were entered into the chip using a software application on a computer.

15. SUBJECT TERMS
RF-MEMS, reliability, lifetime test

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Tony Ivanov
Unclassified	Unclassified	Unclassified	UU	24	19b. TELEPHONE NUMBER (*Include area code*) (301) 394-3568

Contents

List of Figures

1. Introduction

Analyzing Microelectromechanical System (MEMS) switch behavior is a new field of study, which guides the production of MEMS switches. One issue of primary importance is the degradation of contact resistance over the lifetime of the device and whether small variations in the resistance initially can be prognostically used to predict future performance. Unfortunately, commercially available equipment is limited by its data collection rate and is not a realistic option to examine every cycle of a switch expected to potentially survive for several trillion cycles. In addition, these tests must be performed on tens to hundreds of switches, necessitating the need for a parallel measurement solution.

To overcome these challenges, a field-programmable gate array (FPGA)-based solution is being developed. The FPGA chip, coupled with a 16 bit dual channel AD/DA converter will serve as the input source to each switch and the measurement system, diagnosing contact performance every cycle. Although customizable to any switch configuration, the purpose of this project was to determine the feasibility of testing single-pole-single-throw (SPST) RF-MEMS switches, which contain four terminals: bias/actuation, RF-input, RF output, and ground. The project was broken down into three phases. First, the algorithms necessary for the system to function, both in terms of testing and meaningful data display, were developed. Next, based on those algorithms, the FPGA implementation options, direct HDL programming and software defined microcontroller usage, were analyzed to determine the appropriate system to reduce implementation complexity while maximizing scalability. Finally, the initial circuit code was created to demonstrate a proof-of-concept system functioning on a single switch.

Switch behavior will be analyzed using a digital circuit synthesized on an FPGA chip. The chip has two outputs: the first output biases the MEMs switch; the second output feeds a digital-to-analog (DAC) converter which provides the drive signal for testing the switch. It will open and close the switch repeatedly at approximately 20 KHz. An external resistor of 10 kΩ will be connected in series with the switch. The analog output voltage across that resistor will be measured (in both open and closed states) and fed into an analog-to-digital converter (ADC). This bit stream will then be fed into the FPGA which will calculate the switch resistance based upon the voltage drop across the external resistor, the voltage drop across the series connection of the external resistor and the switch, and the resistance of the external resistor. All parameters, such as the bias voltage, will be entered into the chip using a software application on a computer. This FPGA-based instrumentation presents a higher frequency measurement capability for MEMs switches than is currently available commercially in the industry. By using low-cost FPGA's, this setup can easily be scaled up into as many as 100 parallel tests that will enable reliability studies that require hundreds of millions of switch cycles.

2. Experimental Procedure

A block diagram of the system is shown in figure 1. The system will use two DAC generated output voltages, one for device bias and one as a forcing voltage, to generate the test signals. The resistance of the RF-MEMS switch contacts is then calculated based on an ADC measurement of the voltage across an external resistor. The Resistance of the switch will be calculated twice every cycle (in both open and closed states) using the circuit diagram shown in figure 2. The FPGA was programmed using VHDL, and a 16-bit dual channel AD/DA converter was utilized. A variety of MEMS switches can be experimented on as DUTS. Each switch can undergo up to 10 billion cycles. The apparatus will send resistance values and number of completed cycles at a set rate to memory. Memory will be provided by a computer connected to the apparatus by Ethernet.

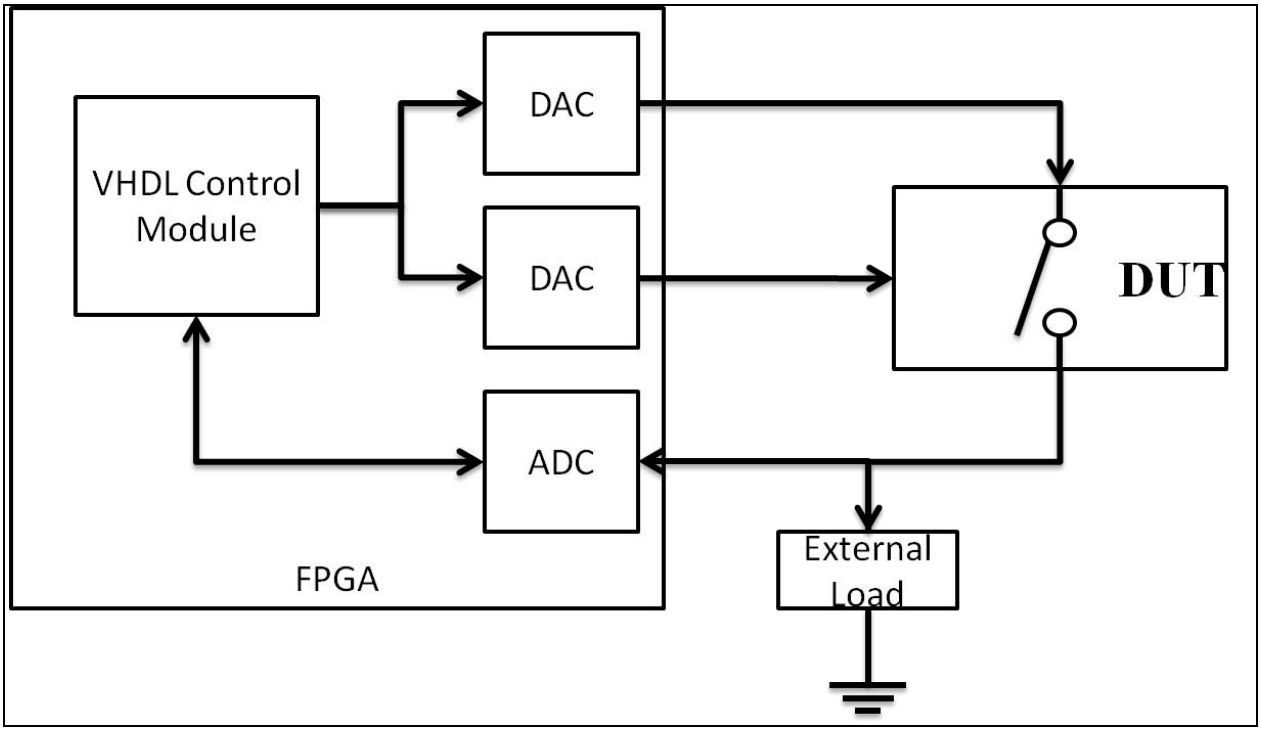

Figure 1. Block diagram of the reliability test system.

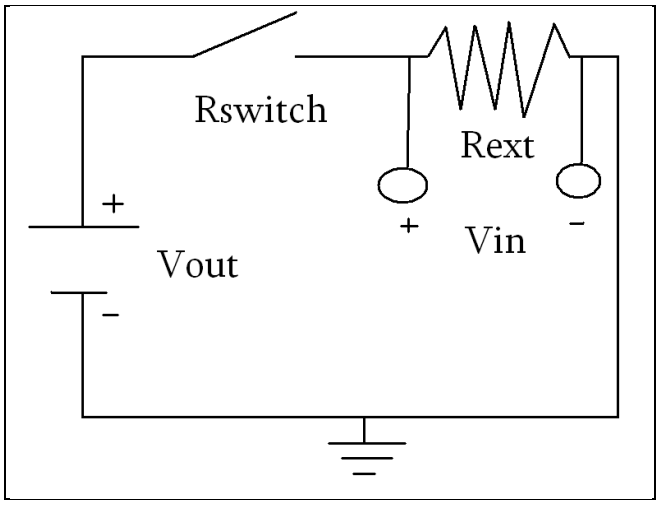

Figure 2. The circuit diagram used to extract the contact resistance value.

The apparatus will look for two types of general failures: temporary and permanent. Both types of failures have two possible modes: stuck open and stuck closed. Once a temporary failure is detected, the switch resistance and number of cycles will be sampled. After crossing a set limit of consecutive temporary failures, the switch will be considered as having permanently failed. A common MEMS switch along with signal contact locations is shown in figure 3.

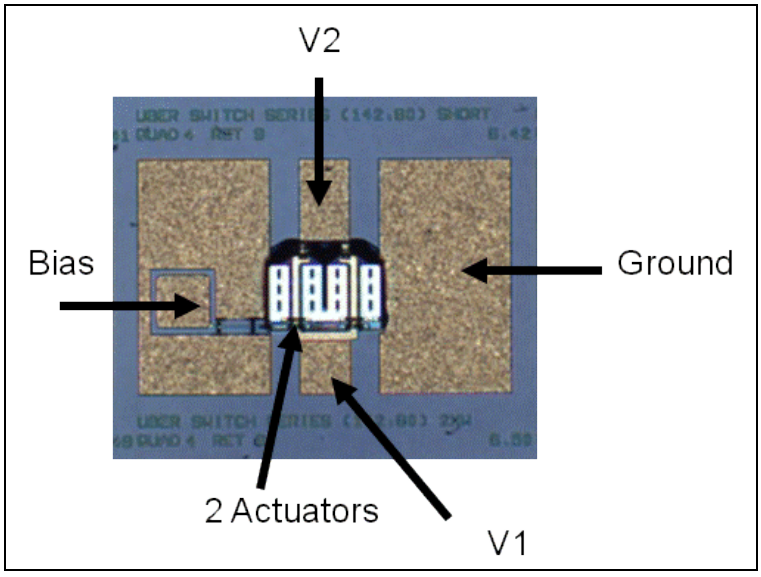

Figure 3. Structure of a typical RF-MEMS switch.

3. Switch Contact Resistance Derivation

Rext: external resistor (10 kΩ)

Rswitch: resistance of closed or open switch

Vout: output voltage signal

Vin: voltage signal across external resistor

Derivation of formula for equating Rswitch:

Vout = i(Rswitch + Rext)

Vin = i(Rext)

i = Vin/Rext

Vout = iRswitch + Vin

Vout = (Vin/Rext) * Rswitch + Vin

Vout = Vin * (Rswitch/Rext + 1)

Vout/Vin = Rswitch/Rext + 1

(Vout/Vin – 1) * Rext = Rswitch

4. HDL and Software Coding

A software based application was used to prototype the primary VHDL module. The VHDL module was responsible for producing each output voltage and measuring the response of the switch. The waveforms generated by the VHDL module, along with the expected responses, are shown in figure 4. In the final module, each input voltage will be converted to a 16-bit array, which will be sent to the storage memory via Ethernet. The module and four finite state machines will constitute the synthesized portion of the FPGA chip. The four finite state machines will consist of two other VHDL modules and two C++ scripts contained inside a synthesized microprocessor. They will allow communication between the computer and the primary module. The five finite state machines will be connected in the following order: primary module, FIFO, check FIFO (C++), transfer data, and check transfer data (C++). The FIFO module uses the first in first out system to take individual sets of data, store them, and then transfer them in segments. The transfer data module completes the process.

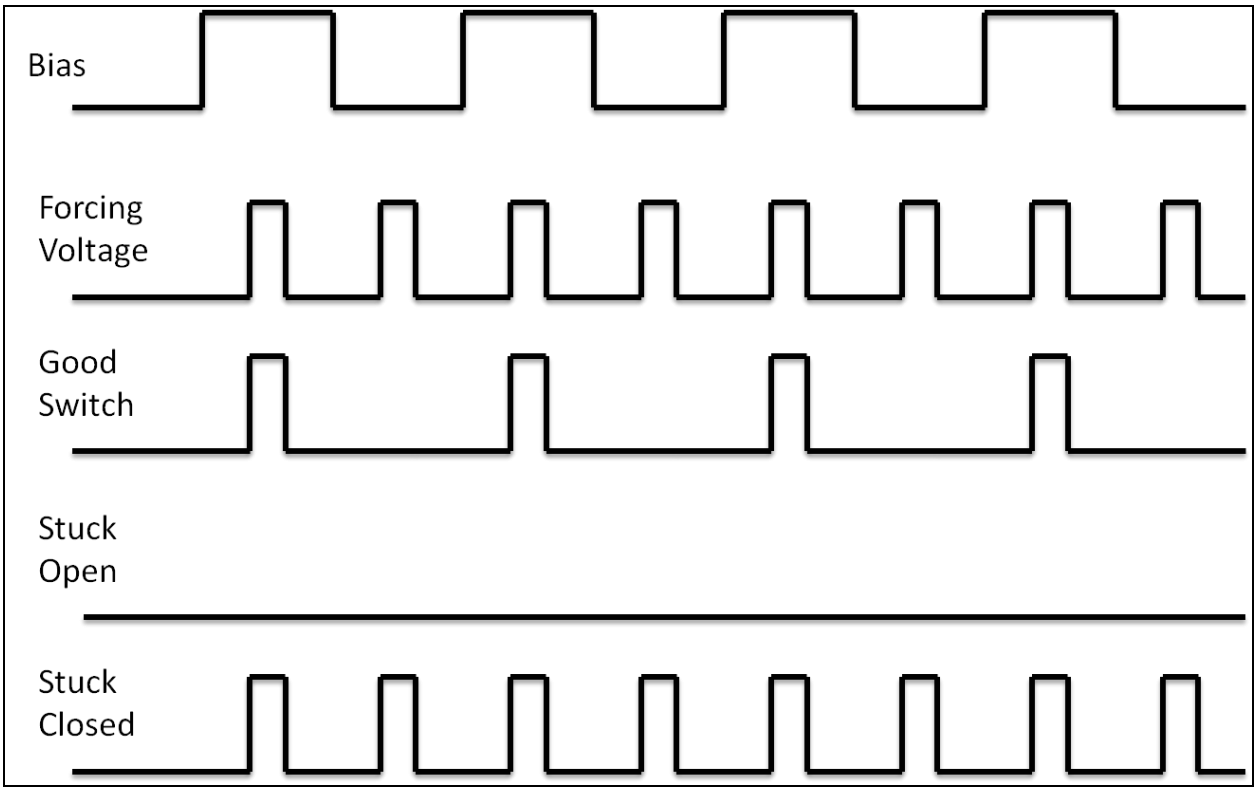

Figure 4. FPGA generated waveforms and the expected output for a good, stuck open and stuck closed switch.

5. Results and Discussion

The apparatus will open and close the switch repeatedly at approximately 20 KHz. The analog output voltage of the switch will be measured (in both open and closed states) and fed into an ADC. This bit stream will then be fed into the FPGA which will calculate the switch resistance based upon the voltage drop across the resistor and switch. All parameters such as the bias voltage will be entered into the chip using a software application on a computer.

6. Summary and Conclusions

The FPGA chip has the available space for the VHDL coded digital circuitry.

7. Suggested Future Work

By using low-cost FPGA’s this setup can easily be scaled up into as many as 100 parallel tests that will enable reliability studies that require hundreds of millions of switch cycles.

Appendix

Software based application used to enter parameters into FPGA (C++):

```
#include <iostream.h>


//parameters entered
int main() {
 double  potential_out_real, bias_switch_real;
 int cycle_sampling_rate;
 int catastophic_failure_stuck_open_count_limit;
 int catastophic_failure_stuck_closed_count_limit;
 int bias_switch_binary[16];
 int potential_out_binary[16];

 cout << "Welcome to Switch Test\n";
 cout << "Enter the following values:\n";
 cout << "cycle_sampling_rate: ";
 cin >> cycle_sampling_rate;
 cout << "catastophic_failure_stuck_open_count_limit: ";
 cin >> catastophic_failure_stuck_open_count_limit;
 cout << "catastophic_failure_stuck_closed_count_limit: ";
 cin >> catastophic_failure_stuck_closed_count_limit;
 cout << "output_voltage: ";
 cin >> potential_out_real;
 cout << "bias_switch_voltage: ";
 cin >> bias_switch_real;
 cout << "values entered\n";



//bias_switch_real and potential_out_real converted to 16bit binary numbers
//first set of 8bits are integer and second set of 8bits are fraction
        int integer_used_for_conversion = bias_switch_real;
        double fraction_used_for_conversion = bias_switch_real - integer_used_for_conversion;

        bias_switch_binary[8] = integer_used_for_conversion%2;
        integer_used_for_conversion = integer_used_for_conversion/2;
        for (int i = 9; i<16; i++)        {
                bias_switch_binary[i] = integer_used_for_conversion%2;
                integer_used_for_conversion = integer_used_for_conversion/2;
        }
```

```
if (fraction_used_for_conversion*2>=1)      {
        fraction_used_for_conversion =      fraction_used_for_conversion*2-1;
        bias_switch_binary[7] = 1;
}
else {
        fraction_used_for_conversion =      fraction_used_for_conversion*2;
        bias_switch_binary[7] = 0;
}
for (int ii = 6; ii>=0; ii--)       {
        if (fraction_used_for_conversion*2>=1)      {
                fraction_used_for_conversion =      fraction_used_for_conversion*2-1;
                bias_switch_binary[ii] = 1;
        }
        else {
                fraction_used_for_conversion =      fraction_used_for_conversion*2;
                bias_switch_binary[ii] = 0;
        }
}


integer_used_for_conversion = potential_out_real;
fraction_used_for_conversion = potential_out_real - integer_used_for_conversion;
potential_out_binary[8] = integer_used_for_conversion%2;
integer_used_for_conversion = integer_used_for_conversion/2;
for (int n = 9; n<16; i++)      {
        potential_out_binary[n] = integer_used_for_conversion%2;
        integer_used_for_conversion = integer_used_for_conversion/2;
}


if (fraction_used_for_conversion*2>=1)      {
        fraction_used_for_conversion =      fraction_used_for_conversion*2-1;
        potential_out_binary[7] = 1;
}
else {
        fraction_used_for_conversion =      fraction_used_for_conversion*2;
        potential_out_binary[7] = 0;
}
for (int nn = 6; nn>=0; nn--)   {
        if (fraction_used_for_conversion*2>=1)      {
                fraction_used_for_conversion =      fraction_used_for_conversion*2-1;
                potential_out_binary[nn] = 1;
        }
        else {
                fraction_used_for_conversion =      fraction_used_for_conversion*2;
```

```
                    potential_out_binary[nn] = 0;
                }
        }

//insert code: transfer the three integer parameters and two bit arrays into FPGA


}
```

Primary VHDL module (VHDL):

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY switch_test_tb IS
END switch_test_tb;

ARCHITECTURE behavior OF switch_test_tb IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT switch_test
   PORT(
        clock : IN  std_logic;
        bias : OUT  std_logic_vector(7 downto 0);
        voltage_out : OUT  std_logic_vector(7 downto 0);
        voltage_in : IN  std_logic_vector(7 downto 0)
       );
   END COMPONENT;


  --Inputs
  signal clock : std_logic := '0';
  signal voltage_in : std_logic_vector(7 downto 0) := (others => '0');

          --Outputs
  signal bias : std_logic_vector(7 downto 0);
  signal voltage_out : std_logic_vector(7 downto 0);

  -- Clock period definitions
  constant clock_period : time := 10 ns;

BEGIN

          -- Instantiate the Unit Under Test (UUT)
  uut: switch_test PORT MAP (
         clock => clock,
         bias => bias,
         voltage_out => voltage_out,
         voltage_in => voltage_in
        );
```

```
  -- Clock process definitions
  clock_process :process
  begin
                clock <= '0';
                wait for clock_period/2;
                clock <= '1';
                wait for clock_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ms.
    wait for 100 ms;

    wait for clock_period*10;

    -- insert stimulus here

    wait;
  end process;

END;
```

FIFO (VHDL):

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity fifo is

port (clock: in std_logic);

end fifo;

architecture Behavioral of fifo is

type state_type is (s1,s2,s3,s4,s5);
signal state : state_type:=s1;
signal write_enable,write_now, write_acknoledge: std_logic ;
begin

sequential_network: process (clock, write_acknoledge)

begin
if clock'event and clock = '1' then
case state is

when s1 =>
--initial state
        state <= s2;
when s2 =>
--insert code: data on bus
        state <= s3;
when s3 =>
--set write_now to '1'
        write_now <= '1';
        state <= s4;
when s4 =>
--insert code: wait for write_acknoledge
        if write_acknoledge = '1' then
                state <= s5;
```

```
        else
                state <= s4;
        end if;
when s5 =>
--set write_now to '0'
        write_now <= '0';
        state <= s1;
when others =>
        null;
end case;
end if;
end process;

write_enbl_combinational_logic: process (write_now, write_acknoledge)
begin
write_enable <= (not write_acknoledge) or write_now;
end process;
end Behavioral;
```

Check FIFO (C++):

```
#include <iostream.h>

int main() {
	bool empty = true;
	char state = '1'; //'1','2','3',or '4'
	switch (state) {
	case '1':	//initial state
		state = '2';
		break;
	case '2':	//insert code: check fifo (VHDL module)
		if (empty)
			state = '1';
		else
			state = '3';
		break;
	case '3':	//inset code: get data
		state = '4';
		break;
	case '4':	//insert code: store data locally
		state = '2';
		break;
	default:
		break;
	}
}
```

Transfer Data (VHDL):

```
    library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity transfer_data is

port (clock, x: in std_logic;
                z: out std_logic);
end transfer_data;

architecture Behavioral of transfer_data is

type state_type is (s1,s2,s3);
signal state : state_type:=s1;
begin

sequential_network: process (clock)

begin
if clock'event and clock = '1' then
case state is

when s1 =>
        --initial state
        if x = '0' then
                state <= s1;
        else
                state <= s2;
        end if;
when s2 =>
        --insert code: UDP connection
        if x = '0' then
                state <= s2;
        else
                state <= s3;
        end if;
when s3 =>
```

```
        --insert code: transfer data
        if x = '0' then
                state <= s3;
        else
                state <= s1;
        end if;

when others =>
        null;
end case;
end if;
end process;
end Behavioral;
```

Check transfer data (C++):

```
#include <iostream.h>

int main() {
	bool incoming_connection_received = false;
	char state = '1'; //'1','2','3',or '4'
	switch (state) {
	case '1':	//initial state
		if (incoming_connection_received)
			state = '2';
		else
			state = '1';
		break;
	case '2':	//insert code: make connection for transfer
		state = '3';
		break;
case '3':	//inset code: get data
		state = '4';
		break;
	case '4':	//insert code: display and store
		state = '1';
		break;
	default:
		break;
	}
}
```

NO. OF COPIES	ORGANIZATION
1 ELEC	ADMNSTR DEFNS TECHL INFO CTR ATTN DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1 CD	OFC OF THE SECY OF DEFNS ATTN ODDRE (R&AT) THE PENTAGON WASHINGTON DC 20301-3080
1	US ARMY RSRCH DEV AND ENGRG CMND ARMAMENT RSRCH DEV & ENGRG CTR ARMAMENT ENGRG & TECHNLGY CTR ATTN AMSRD AAR AEF T J MATTS BLDG 305 ABERDEEN PROVING GROUND MD 21005-5001
1	PM TIMS, PROFILER (MMS-P) AN/TMQ-52 ATTN B GRIFFIES BUILDING 563 FT MONMOUTH NJ 07703
1	US ARMY INFO SYS ENGRG CMND ATTN AMSEL IE TD A RIVERA FT HUACHUCA AZ 85613-5300
1	COMMANDER US ARMY RDECOM ATTN AMSRD AMR W C MCCORKLE 5400 FOWLER RD REDSTONE ARSENAL AL 35898-5000
1	US GOVERNMENT PRINT OFF DEPOSITORY RECEIVING SECTION ATTN MAIL STOP IDAD J TATE 732 NORTH CAPITOL ST NW WASHINGTON DC 20402
1	US ARMY RSRCH LAB ATTN RDRL CIM G T LANDFRIED BLDG 4600 ABERDEEN PROVING GROUND MD 21005-5066

NO. OF COPIES	ORGANIZATION
5	US ARMY RSRCH LAB ATTN IMNE ALC HRR MAIL & RECORDS MGMT ATTN RDRL CIM L TECHL LIB ATTN RDRL CIM P TECHL PUB ATTN RDRL SER E T IVANOV (2 HCS) ADELPHI MD 20783-1197

TOTAL: 13 (1 ELEC, 1 CD, 11 HCS)